Help Logout Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

Search Results -

Terms	Documents
L2 and L1	12

1376
3.26.03

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L3

Refine Search

Recall Text Clear

Search History

DATE: Wednesday, March 26, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L3	L2 and L1	12	L3
L2	((707/200 |707/201 |707/202 |707/203 |707/204 |707/205 |707/206)!.CCLS.)	3546	L2
L1	dump$ and ((partition$ or divid$ or cell$2) near2 (database$2 or table$2))	242	L1

END OF SEARCH HISTORY

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 10 of 12 returned.

☐ 1. Document ID: US 6240428 B1

L3: Entry 1 of 12 File: USPT May 29, 2001

US-PAT-NO: 6240428
DOCUMENT-IDENTIFIER: US 6240428 B1

TITLE: Import/export and repartitioning of partitioned objects

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 2. Document ID: US 6192365 B1

L3: Entry 2 of 12 File: USPT Feb 20, 2001

US-PAT-NO: 6192365
DOCUMENT-IDENTIFIER: US 6192365 B1

TITLE: Transaction log management in a disconnectable computer and network

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 3. Document ID: US 6154748 A

L3: Entry 3 of 12 File: USPT Nov 28, 2000

US-PAT-NO: 6154748
DOCUMENT-IDENTIFIER: US 6154748 A

TITLE: Method for visually mapping data between different record formats

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 4. Document ID: US 6105033 A

L3: Entry 4 of 12 File: USPT Aug 15, 2000

US-PAT-NO: 6105033
DOCUMENT-IDENTIFIER: US 6105033 A

TITLE: Method and apparatus for detecting and removing obsolete cache entries for enhancing cache system operation

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | Image

☐ 5. Document ID: US 6065017 A

L3: Entry 5 of 12 File: USPT May 16, 2000

US-PAT-NO: 6065017
DOCUMENT-IDENTIFIER: US 6065017 A

TITLE: Apparatus and method for identifying and recovering from database errors

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

---

☐ 6. Document ID: US 5991771 A

L3: Entry 6 of 12 File: USPT Nov 23, 1999

US-PAT-NO: 5991771
DOCUMENT-IDENTIFIER: US 5991771 A

TITLE: Transaction synchronization in a disconnectable computer and network

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

---

☐ 7. Document ID: US 5878434 A

L3: Entry 7 of 12 File: USPT Mar 2, 1999

US-PAT-NO: 5878434
DOCUMENT-IDENTIFIER: US 5878434 A

TITLE: Transaction clash management in a disconnectable computer and network

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

---

☐ 8. Document ID: US 5794229 A

L3: Entry 8 of 12 File: USPT Aug 11, 1998

US-PAT-NO: 5794229
DOCUMENT-IDENTIFIER: US 5794229 A

TITLE: Database system with methodology for storing a database table by vertically partitioning all columns of the table

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

---

☐ 9. Document ID: US 5794228 A

L3: Entry 9 of 12 File: USPT Aug 11, 1998

US-PAT-NO: 5794228
DOCUMENT-IDENTIFIER: US 5794228 A

TITLE: Database system with buffer manager providing per page native data compression and decompression

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments KWIC | Draw Desc | Image

☐ 10. Document ID: US 5721918 A

L3: Entry 10 of 12 File: USPT Feb 24, 1998

US-PAT-NO: 5721918
DOCUMENT-IDENTIFIER: US 5721918 A

TITLE: Method and system for fast recovery of a primary store database using selective recovery by data type

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments KWIC | Draw Desc | Image

Generate Collection Print

Terms	Documents
L2 and L1	12

Display Format: - Change Format

Previous Page Next Page

WEST

Generate Collection | Print

Search Results - Record(s) 11 through 12 of 12 returned.

☐ 11. Document ID: US 5204958 A

L3: Entry 11 of 12 File: USPT Apr 20, 1993

US-PAT-NO: 5204958
DOCUMENT-IDENTIFIER: US 5204958 A

TITLE: System and method for efficiently indexing and storing a large database with high data insertion frequency

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

☐ 12. Document ID: US 5043871 A

L3: Entry 12 of 12 File: USPT Aug 27, 1991

US-PAT-NO: 5043871
DOCUMENT-IDENTIFIER: US 5043871 A

TITLE: Method and apparatus for database update/recovery

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC | Draw Desc | Image

Generate Collection | Print

Terms	Documents
L2 and L1	12

Display Format: - Change Format

Previous Page Next Page

WEST

Help | Logout | Interrupt

Main Menu | Search Form | Posting Counts | Show S Numbers | Edit S Numbers | Preferences | Cases

3/26/03

Search Results -

Terms	Documents
L7 and (partition$ near2 table$2)	3

Database: US Patents Full-Text Database; US Pre-Grant Publication Full-Text Database; JPO Abstracts Database; EPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search: dump$ and partition$ and table$2

Refine Search | Recall Text | Clear

Search History

DATE: Wednesday, March 26, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT,PGPB,JPAB,EPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L8	L7 and (partition$ near2 table$2)	3	L8
L7	(dump$ near2 file$2) and partition$	83	L7
L6	L5 not L4	10	L6
L5	(dump$ near20 file$2) and (partition$ near20 table$2)	13	L5
L4	(dump$ near2 file$2) and (partition$ near2 table$2)	3	L4
L3	("dump file") and ("partitions table")	1	L3
L2	("dump file") and (partitions table)	144	L2
L1	"dump file"	271	L1

END OF SEARCH HISTORY

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 3 of 3 returned.

☐ 1. Document ID: US 20030009295 A1

L8: Entry 1 of 3 File: PGPB Jan 9, 2003

PGPUB-DOCUMENT-NUMBER: 20030009295
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20030009295 A1

TITLE: System and method for retrieving and using gene expression data from multiple sources

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments KWIC | Draw Desc | Image

☐ 2. Document ID: US 6240428 B1

L8: Entry 2 of 3 File: USPT May 29, 2001

US-PAT-NO: 6240428
DOCUMENT-IDENTIFIER: US 6240428 B1

TITLE: Import/export and repartitioning of partitioned objects

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments KWIC | Draw Desc | Image

☐ 3. Document ID: US 5511190 A

L8: Entry 3 of 3 File: USPT Apr 23, 1996

US-PAT-NO: 5511190
DOCUMENT-IDENTIFIER: US 5511190 A

TITLE: Hash-based database grouping system and method

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments KWIC | Draw Desc | Image

Generate Collection | Print

Terms	Documents
L7 and (partition$ near2 table$2)	3

Display Format: - Change Format

Previous Page Next Page

Search Results

Search Results for: **[(DDL <or> (data <near> description <near> language)) <paragraph> (partition* <near> table) <and> (dump <near> file)]**
Found **4** of **126,861 searched.**

Search within Results

GO > Advanced Search

> Search Help/Tips

Sort by: Title Publication Publication Date Score Binder

Results 1 - 4 of 4 short listing

1 Toward user sharing of the microprogramming level under UNIX on the Perkin-Elmer 3220 89%

J. Eric Roskos , Robert I. Winner

Proceedings of the 14th annual workshop on Microprogramming December 1981

A master/slave model of writable control store is presented which is claimed to be a better representation of the operating system view of control store than models which more accurately portray the physical reality. Reported work includes the completed development of UNIX tools: an assembler generating relocatable, linkable microcode; a linking, relocating loader; a fast absolute loader; an interactive hard debugging aid allowing the setting of breakpoints; and alterations of the ...

2 Software support for the Yorktown Simulation Engine 84%

E. Kronstadt , G. Pfister

Proceedings of the nineteenth design automation conference January 1982

The Yorktown Simulation Engine (YSE) is a special-purpose, highly-parallel programmable machine for the gate-level simulation of logic. The YSE has been designed and is being constructed at the IBM T. J. Watson Research Center. It can simulate up to one million gates at a speed of over two billion gate simulations per second; it is estimated that the IBM 3081 processor could have been simulated on the YSE at a rate of 1000 instructions per second. This is far beyond the capabilities of exis ...

3 Converting thread-level parallelism to instruction-level parallelism via simultaneous multithreading 47%

Jack L. Lo , Joel S. Emer , Henry M. Levy , Rebecca L. Stamm , Dean M. Tullsen , S. J. Eggers

ACM Transactions on C mputer Systems (TOCS) August 1997

Volume 15 Issue 3

To achieve high performance, contemporary computer systems rely on two forms of

parallelism: instruction-level parallelism (ILP) and thread-level parallelism (TLP). Wide-issue super-scalar processors exploit ILP by executing multiple instructions from a single program in a single cycle. Multiprocessors (MP) exploit TLP by executing different threads in parallel on different processors. Unfortunately, both parallel processing styles statically partition processor resources, thus preventing t ...

4 A view of a user-oriented production test data generation system 10%

Peter S. Bottorff , Robert A. Rasmussen

Proceedings of the June 1970 design automation workshop on Design automation

June 1970

Present methods of production test data generation for dense logic circuit packages are costly and time-consuming. Necessary human intervention can be performed only between batch computer runs. Speed and cost improvements can be made with a system that provides the user with real-time information on the consequences of his decisions. This paper presents design considerations for a system which meets this need.

Results 1 - 4 of 4 **short listing**

Search Results

Search Results for: **[(DDL <or> (data <near> description <near> language)) <and> ((partition* <near> table) <paragraph> (dump <near> file))]**
Found **2** of **126,861 searched.**

Search within Results

GO > Advanced Search

> Search Help/Tips

Sort by: **Title** **Publication** **Publication Date** **Score** Binder

Results 1 - 2 of 2 **short listing**

1 Validation coverage analysis for complex digital designs 50%

Richard C. Ho , Mark A. Horowitz
Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design January 1997

In interactive behavioral synthesis, the designer can control the design process at every stage, including modifying the schedule of the design to improve its performance. In this paper, we present a methodology for performance optimization in interactive behavioral synthesis. Also proposed are several quality metrics and hints that can assist the user in utilizing the proposed methodology. When the user is optimizing the performance of the design, one important decision is the selection of a cl ...

2 Real-time programming in PORTAL 6%

R. Schild , H. Lienhard
ACM SIGPLAN Notices April 1980
Volume 15 Issue 4

Results 1 - 2 of 2 **short listing**

h c g e cf c

Search Results

Search Results for: **[dump* <paragraph> partition* <paragraph> (DDL <or> (data <near> description <near> language))]**
Found **2** of **126,861 searched.**

Search within Results

GO > Advanced Search

> Search Help/Tips

Sort by: **Title** **Publication** **Publication Date** **Score** Binder

Results 1 - 2 of 2 **short listing**

1 Introductory computer science courses a modular design 100%

Ivan M. Rosenberg

Proceedings of the ACM SIGCSE-SIGCUE technical symposium on Computer science and education February 1976

Volume 2 , 8 Issue SI , 1

This paper proposes a set of assumptions about learning in general, followed by a set of assumptions concerning ICS courses in particular. Using these as a foundation, an ICS course content and structure is developed which clearly shows the relationships between the various branches of computer science, encourages a logical presentation, and is modular and hierarchical, permitting use for a wide variety of audiences and course objectives. Since content organization is only one aspect of a c ...

2 Validation coverage analysis for complex digital designs 100%

Richard C. Ho , Mark A. Horowitz

Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design January 1997

In interactive behavioral synthesis, the designer can control the design process at every stage, including modifying the schedule of the design to improve its performance. In this paper, we present a methodology for performance optimization in interactive behavioral synthesis. Also proposed are several quality metrics and hints that can assist the user in utilizing the proposed methodology. When the user is optimizing the performance of the design, one important decision is the selection of a cl ...

Results 1 - 2 f 2 **short listing**

h c g e cf c

Search Results

Search Results for: [dump* <PARAGRAPH> file <PARAGRAPH> partition* <PARAGRAPH> (database OR table)]
Found 3 of 107,580 searched. → Rerun within the Portal

Search within Results

Results 1 - 3 of 3 **short listing**

1 Audit trail compaction for database recovery 100%
John Kaunitz , Louis van Ekert
Communications of the ACM July 1984
Volume 27 Issue 7
Total elapsed recovery time from disk-based database corruption can be shortened by reprocessing the audit trail off-line and thereby avoiding excessive resource utilization penalties. Using a bit map, the audit trail is compacted by eliminating irrelevant or superseded records. The compacted trail is then partitioned, and the partitions are processed in parallel.

2 Modeling design constraints and biasing in simulation using BDDs 100%
Jun Yuan , Kurt Shultz , Carl Pixley , Hillel Miller , Adnan Aziz
Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design November 1999
Constraining and input biasing are frequently used techniques in functional verification methodologies based on randomized simulation generation. Constraints confine the simulation to a legal input space, while input biasing, which can be considered as a probabilistic constraint, makes it easier to cover interesting “corner” cases. In this paper, we propose to use constraints and biasing to form a simulation environment instead of using an expli ...

3 Avoiding unconditional jumps by code replication 100%

Frank Mueller , David B. Whalley

ACM SIGPLAN Notices , Proceedings of the 5th ACM SIGPLAN conference on Programming language design and implementation July 1992

Volume 27 Issue 7

This study evaluates a global optimization technique that avoids unconditional jumps by replicating code. When implemented in the back-end of an optimizing compiler, this technique can be generalized to work on almost all instances of unconditional jumps, including those generated from conditional statements and unstructured loops. The replication method is based on the idea of finding a replacement for each unconditional jump which minimizes the growth in code size. This is achieved by cho ...

Results 1 - 3 of 3 **short listing**

Search Results

Search Results for: [((database OR table) <NEAR> partition*) <paragraph> (dump* <NEAR> file)]
Found 6 of 107,580 searched. → Rerun within the Portal

Search within Results

> Advanced Search > Search Help/Tips

Sort by: Title Publication Publication Date Score Binder

Results 1 - 6 of 6 short listing

1 Audit trail compaction for database recovery 100%
John Kaunitz , Louis van Ekert
Communications of the ACM July 1984
Volume 27 Issue 7
Total elapsed recovery time from disk-based database corruption can be shortened by reprocessing the audit trail off-line and thereby avoiding excessive resource utilization penalties. Using a bit map, the audit trail is compacted by eliminating irrelevant or superseded records. The compacted trail is then partitioned, and the partitions are processed in parallel.

2 Audit considerations in distributed processing systems 100%
James V. Hansen
Communications of the ACM August 1983
Volume 26 Issue 8
Applications of distributed processing networks are proliferating rapidly. It is expected that by the year 2000, distributed networks will be one of the most significant developments to evolve from the computer revolution. Distributed networks are unique in that they bring together concepts of communication, engineering, and computing. From an audit standpoint, the complexities involved in control design and testing are challenging. The auditor needs to b ...

3 Modeling design constraints and biasing in simulation using 100%

BDDs
Jun Yuan , Kurt Shultz , Carl Pixley , Hillel Miller , Adnan Aziz
Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design November 1999
Constraining and input biasing are frequently used techniques in functional verification methodologies based on randomized simulation generation. Constraints confine the simulation to a legal input space, while input biasing, which can be considered as a probabilistic constraint, makes it easier to cover interesting “corner” cases. In this paper, we propose to use constraints and biasing to form a simulation environment instead of using an expli ...

4 Validation coverage analysis for complex digital designs 100%
Richard C. Ho , Mark A. Horowitz
Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design January 1997

5 Does licensing require new access control techniques? 100%
Ralf C. Hauser
Communications of the ACM November 1994
Volume 37 Issue 11

6 Avoiding unconditional jumps by code replication 100%
Frank Mueller , David B. Whalley
ACM SIGPLAN Notices , Proceedings of the 5th ACM SIGPLAN conference on Programming language design and implementation July 1992
Volume 27 Issue 7
This study evaluates a global optimization technique that avoids unconditional jumps by replicating code. When implemented in the back-end of an optimizing compiler, this technique can be generalized to work on almost all instances of unconditional jumps, including those generated from conditional statements and unstructured loops. The replication method is based on the idea of finding a replacement for each unconditional jump which minimizes the growth in code size. This is achieved by cho ...

Results 1 - 6 of 6 **short listing**

L22: Entry 1 of 2 File: USPT Apr 16, 2002

DOCUMENT-IDENTIFIER: US 6374267 B1
TITLE: Database backup system and method utilizing numerically identified files for incremental dumping

Brief Summary Paragraph Right (5):
Another way is (ii) designated logical level back-up. This refers to the data as seen by the user application programs in files or in database tables. Normally, the operating system of the computer will include a file system that does mapping between the physical level and the logical level. On doing the physical level back-up, this would involve making a raw copy from a computer disk to some other storage medium without going through the file system or some other physical to logical interpreter module. Then on the other hand, the back-up using the logical level would involve using such a thing as an interpreter module or some sort of a file system while doing back-up of physical to logical mapping.

Brief Summary Paragraph Right (14):
Actual real world data goes into special logical structures that are used by the Data Management System to store data. The database is designed to map categories of data into suitable structures. For example, the real world data would have a character with a structure called a "data set". An example of this would be a particular person's name. Then, real world data that can serve as an index of a whole data set has a structured name called a "set". This, for example, might be the social security number of any employee. Then there is data that can serve as an index of a data set under a certain condition, and this is called a "subset". This might be an employee's work number, for example. Then, there is data about each instance of a particular category. The structure name for this is "data item". An example of this might be the name and address of the category (person). Then, there is data related to the database as a whole, and this involves a structure called "global data item". An example of this might be the total number of employees in a company. Once there has been identification of the real-world data which is to be stored in the database, it is then necessary to define that data in relationship to the data structures of the data management system that holds data. When this data is defined within "structures", then the data management system and the system software programs an application program that can then understand how to make this data accessible for various inquiries and/or changes. This is done with the Data and Structure Definition Language (DASDL).

Brief Summary Paragraph Right (15):
The Data Management System structures are the building blocks of the Data Management System database. Here, the "data set" has the purpose of storing data pertaining to a data category in a collection of records. A "set" has the purpose of indexing all records in a data set. A "subset" serves the purpose to index some records in a data set according to some given criteria. The "data item" is a structured name which defines a unit of information about a category in a given field (column) of a data set record. A "global data item" serves the purpose of storing a unit of information about the entire database or any of its involved structures. In general discussion about the types of data and the names of data structures, it is often seen that in a relational database, a "data set" is called a "table". A "set" or "subset" is frequently called an "index". A "data item" is often called a "field" or a "column", or is often called by its data name, for example, a project number. "Structures" are made of common file components designated as records and fields.

Brief Summary Paragraph Right (21):

A data set is kept up-to-date in several ways: (i) here, application programs add, change, or delete individual pieces of data or records stored in the data set; (ii) the Database Administrator (DBA) maintains the structure of the data set by keeping the data set within certain maximized limits, by adding, deleting or changing the definition of a data item, creating new sets or subsets, monitoring automatic processes that guard data integrity and creating guard files to enhance the security of the data.

Brief Summary Paragraph Right (23):
A "subset" can be considered identical to a set, except that the subset need not contain a record for every record of the data set. A subset is a file that indexes none, one, several, or all of the records in a data set. The subset structure enables an application program to access only records of a data set that meet a particularly required condition.

Brief Summary Paragraph Right (24):
For example, an application program may compile a list of people who are "managers". Thus, it is seen that the database designer created the "manager" subset. Thus, in order to retrieve a record of managers, the data management system can use the smaller file, that is, the subset, to quickly point to the corresponding records in the larger file which is the data set. As with the set, the subset must also be kept up-to-date.

Brief Summary Paragraph Right (33):
However, when dumping to disk it is only necessary to specify the file title for the entire dump and the number of DUMP files into which the system should place the DUMP.

Detailed Description Paragraph Right (10):
In order to back-up the database files other than using the DUMPS and to have all database files backed-up in case the User needs to re-establish the current database from scratch, there are certain files that are required to be held in reserve and these are:

Detailed Description Paragraph Right (15):
When the user chooses to back-up each data set and its associated sets and subsets using the Incremental or the Accumulated Dump feature, the DUMPSTAMP option needs to be set. The DUMPSTAMP option is reset by default for existing structures. By specifying DUMPSTAMP or DUMPSTAMP=TRUE, will set the option. Setting the DUMPSTAMP to TRUE for existing structures requires a file format reorganization and yet still there is no need for recompilation of the application program.

Detailed Description Paragraph Right (17):
INDEPENDENTTRANS must also be set when DUMPSTAMP is set to TRUE. When Users are performing a reorganization, the DUMPSTAMP option is set to TRUE. This can be accomplished by using the defaults enforced by the DASDL compiler or by explicitly setting the DUMPSTAMP option to TRUE. The DUMPSTAMP option is not designed for partitioned data sets.

Detailed Description Paragraph Right (19):
When the User chooses to back-up each data set and its associated sets and subsets using the Incremental or Accumulated Dump feature, the DUMPSTAMP option needs to be set. The DUMPSTAMP option is reset by default for existing structures. Specifying DUMPSTAMP or setting DUMPSTAMP=TRUE will set the option, thus setting the DUMPSTAMP to true for existing structures will require a file format reorganization, yet at the same time, no recompilation of application programs is necessary. Once the DUMPSTAMP option is set, then resetting it will not force a file format reorganization, and the Incremental and the Accumulated Dump capability stays enabled.

Detailed Description Paragraph Right (32):
FIG. 3A is a drawing indicating features of the Dump-to-Disk data flow. For example, the source disk 40S is seen to have a multitude of separate areas designated 41a through 41m. Each of these areas are transferred to separate Input buffers designated as 43 and 44 which have been expanded from their normal 4,800 words to

65,535 words. Subsequently, the data in the Input buffers 43 and 44 are then respectively transferred to the Output buffers 45a . . . 45n and 46a . . . 46x thence to the destination Dump file 40d. The Output buffers 45 are then placed in blocks designated 47, while the Output buffers 46 are then placed in blocks 48 within the destination Dump file 40d. It is also noted that there is now provided an increased default block size from the normal 903 words up to 20,040 words. A more efficient data transfer operation is now enabled since larger block sizes of words can be transferred per any given single command operation.

Detailed Description Paragraph Right (36):
Subsequently, the information will be placed on the destination DUMP file on Tape 50. On Tape 50, there will be a series of blocks designated 47a . . . 47n which derive from the buffers 45a . . . 45n. Likewise, the DUMP file on Tape 50 will also have a number of blocks designated 48a . . . 48x which have been derived from the Output buffers 46a . . . 46x.

Detailed Description Paragraph Type 0 (35):
DUMP OUTPUT LISTING: Following the execution of each dump command, an output listing will be sent to a session printer backup file so the user can have hard copies of the dump directory and files that were included in the dump operation.

Detailed Description Paragraph Type 0 (41):
EXCLUDE PARAMETER: This is for the DMUTILITY Dump command which has the purpose of excluding one or more structures from the database dump. This new feature provides flexibility to the current Dump command and is especially helpful when a small percent of the structures are being excluded from the Dump operation. This command is supported for the DMUTILITY Dumps with all the files selected, and it allows user to select one or more disjoint data sets and all of the structures associated with it (sets, subsets, embedded structures, etc.) which are to be excluded from each dump session.

Detailed Description Paragraph Type 0 (43):
GARBAGE COLLECTION: In Data Management System II (DMSII), the process of consolidating deleted or unused space in data sets, sets, and subsets, and returning this space to the system.

Detailed Description Paragraph Type 0 (49):
KEY: (1) A field used to locate or identify a record in an indexed file. (2) In COBOL, a data item that identifies the location of a record, or a group of data items that identifies the ordering of data. (3) In Data Management System II (DMSII), a field that serves as a retrieval key for a set or subset.

Detailed Description Paragraph Type 0 (52):
LABEL: (1) The first 28 sectors on a disk, on which information about the disk is stored. This information includes the family name and serial number, the Master Available Table (MAT), the family index number, information about the family's base pack, and a pointer to the system directory if the disk contains a directory. (2) In RPG and ALGOL, a name that identifies either a point in the Calculation Specifications where a GOTO operation branches or the beginning of a subroutine.

Detailed Description Paragraph Type 0 (53):
LIBRARY: (1) A collection of one or more named routines or entry points that are stored in a file and can be called by other programs. (2) A program that exports objects for use by user programs.

Detailed Description Paragraph Type 0 (54):
LIBRARY DIRECTORY: A memory structure associated with a library process stack that describes the objects exported by the library process.

Detailed Description Paragraph Type 0 (75):
VERIFYDUMP A command which will check the dump file against errors such as block checksum, error block sequencing error, and I/O errors. If a problem has been detected, an error message will be displayed.

Detailed Description Paragraph Type 0 (76):

DUMPSTAMP: When the DUMPSTAMP option is set, a new word is allocated to store the transtamp of each data block. This attribute is valid for data sets, sets and subsets. The Incremental and Accumulated Dump feature in DMUTILITY uses the transtamp value to detect when data blocks have been modified. Then this information is used to determine whether each data block will be included in the

Detailed Description Paragraph Type 0 (77):
DUMPSTAMP OPTION: When the DUMPSTAMP option is set, a new word is allocated to store the last update transtamp of each data block. This attribute is valid for data sets, sets, and subsets. The Incremental and Accumulated Dump feature in DMUTILITY uses the transtamp value to detect when data blocks have been modified. This information is then used to determine whether each data block will be included in the Dump.

Detailed Description Paragraph Type 0 (78):
INCREMENTAL OPTION: The purpose and use of the Incremental option is to back-up all data sets, sets, and subsets which have been modified since the last Incremental, Accumulated, or full Dump has been performed. All structures will have the DUMPSTAMP option set in DASDL and will contain an extra word for storing information, and this trans-stamp is used by the DMUTILITY to decide whether each data block will be included in the Incremental Dump process.

Detailed Description Paragraph Type 0 (80):
ACCUMULATED OPTION: The purpose and use of the ACCUMULATED option is to back-up all data sets and subsets, which have been modified since the last full Dump had been performed. All structures which have the DUMPSTAMP option set in DASDL will contain an extra word for storing information and this transtamp is used by DMUTILITY to decide whether each data block will be included the ACCUMULATED Dump process.

WEST

End of Result Set

Generate Collection | Print

L9: Entry 2 of 2 File: USPT Aug 27, 1991

DOCUMENT-IDENTIFIER: US 5043871 A
TITLE: Method and apparatus for database update/recovery

Brief Summary Text (5):
(1) A database is constructed of pages each constituting a logical area unit. Two page tables including a current version and a backup version are provided, the table indicating a correspondence between pages and slots in the database storage medium in which the contents of pages are stored. The tables are stored in non-volatile medium such as on a disk. The page table itself is usually divided into plural blocks. Transfer between a main storage and the non-volatile medium is executed in units of blocks.

Brief Summary Text (24):
According to another embodiment of the present invention, a database is divided into a plurality of sub-areas Backup files each having the content of each sub-area and differential files each having updated information of each sub-area sequentially recorded, are provided to thus make it possible to obtain and analyze in parallel the update journals.

Detailed Description Text (16):
First, the structure and operation of a first embodiment will be described. FIG. 1 shows the structure of a first embodiment wherein the gist of the present invention is illustrated. Referring now to FIG. 1, connected to a CPU/memory 10 are a database storage medium 50 such as a magnetic disk, a non-volatile storage medium 100 storing database control information such as a backup version page table 80 and a backup version slot use map 90, and a magnetic disk (or magnetic tape) serving as a medium of a journal file 70. The database area 60 in the database storage medium 50 is divided into fixed length blocks called slots. A logical space of the database to be referred to and updated by a transaction is divided into fixed length areas called pages. A slot is used to store the contents of a page, both the slot and the page have the same size (for example, 4096 bytes). Each slot can be referred to or updated by a single input/output operation. FIG. 2 illustrates the backup version page table 80 and the backup version slot use map 90. The backup version page table 80 is updated at a checkpoint to reflect the contents of the current version page table 30 at that time. Particularly, the backup version page table 80 is set with a slot number representative of the storage location in the database area 60 of the latest contents of each page at a latest checkpoint. The backup version slot use map 90 is updated at a checkpoint similar to the case of the backup version page table 80, the backup version slot map 90 being a bit map for indicating use or non-use of each slot in the database 60 in accordance with the current version page table 30 at that time. In particular, in the backup version slot use map 90, 1 is set at the slot storing the latest contents of each page at a latest checkpoint, and 0 is set at the other slots. The backup version page table 80 and the backup version slot use map 90 are constructed of a plurality of fixed length (e.g., 512 bytes) blocks 81 and 91, respectively. Each block can be referred to or updated by a single input/output operation. If a failure occurs while the backup version page table 80 is under a write operation, logical integrity of the database is not ensured. To eliminate this problem, the backup version page table 80 is duplicated such that after a first write operation is completed in a normal manner, a next write operation continues with respect to the other backup version page table. If the first write operation terminates abnormally, the other backup version page table is

used to execute a restore process as of an ordinary system failure so that logical integrity of the database system can be ensured. Consequently, in this embodiment, it is assumed that a write operation of the backup version page table 80 is always completed in a normal manner. The backup version page table 80 and the backup version slot use map 90 may be located in the database storage medium 50.

Detailed Description Text (162):
To solve the problems (1) and (2), a database is divided into a plurality of sub-areas. Backup files each having the contents of each sub-area and differential files each having updated information of each sub-area sequentially recorded, are provided.

Detailed Description Text (172):
First, the structure of the system will be described. FIG. 13 illustrates the feature and system structure of this embodiment. A database 520 and differential files 530 are coupled to a CPU/memory 510. Alternate differential files 540 and backup files 550 are also included in the system structure. As shown in FIG. 13, the differential files, the alternate differential files, and the backup files are recorded in medium such as magnetic tapes capable of off-line recording information. The database is divided into n sub-areas Dk (k=1, . . . , n). Although each sub-area is shown as a separate medium in FIG. 13 for simplicity purpose, the sub-areas are not necessarily limited to such arrangement. The backup file, differential file, and alternate differential file corresponding to a sub-area Dk are respectively represented by Bk, Sk, and Sk' (k=1, 2, . . . , n). Differential files Sk and Sk' are used alternately as described later.

Detailed Description Text (187):
If the database is divided into n sub-areas, the efficiency of fetching database update journal can be improved by 1 to n times as good as a conventional method. For example, assume that the database update journal is fetched into a magnetic tape (effective availability during transfer is 50%) at a transfer speed of 3M byte/sec and that database update is executed equally by four (n=4) sub-areas. If the amount of database update journal per transaction is 2 KB, the upper limit of process capability becomes 3000 transaction/sec as compared with a conventional 750 (=3.times.0.5.times.100/2) transaction/sec.

CLAIMS:

5. A method according to claim 4, wherein at a time of recovery of said current version page table when a system failure occurs, by using the journal record in said journal file representative of the contents of change of said current version page table associated with the database update by a transaction, a latest, effectively stored checkpoint dump of said current version page table in said checkpoint dump file, and said backup version page table in said non-volatile storage medium, said current version page table is recovered such that only the change of said current version page table associated with the database update by the transaction whose journal record indicates the completion of the transaction is restored.

10. A method according to claim 9, wherein at a time of recovery of said current version page table when a system failure occurs, by using the journal record in said journal file representative of the updated contents of said current version page table associated with the database update by a transaction, a latest, effectively stored checkpoint dump of said current version page table in said checkpoint dump file, and said backup version page tables in said non-volatile storage medium, said current version page table is recovered such that only the change of said current version page table associated with the database update by the transaction whose journal record indicates the completion of the transaction is restored.